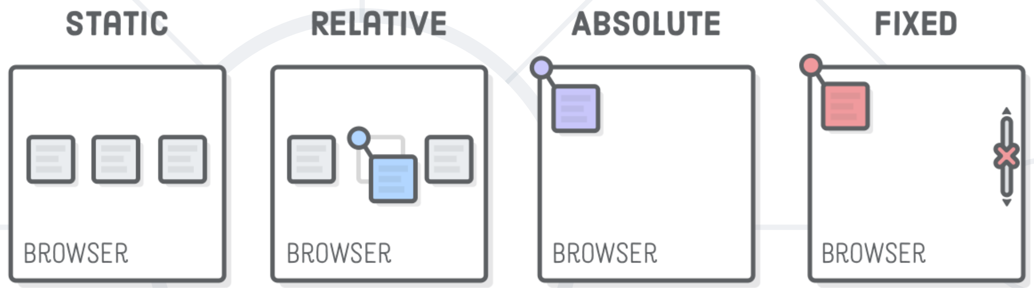
POSITION & FLOAT



SoftUni Team

Technical Trainers

Software University

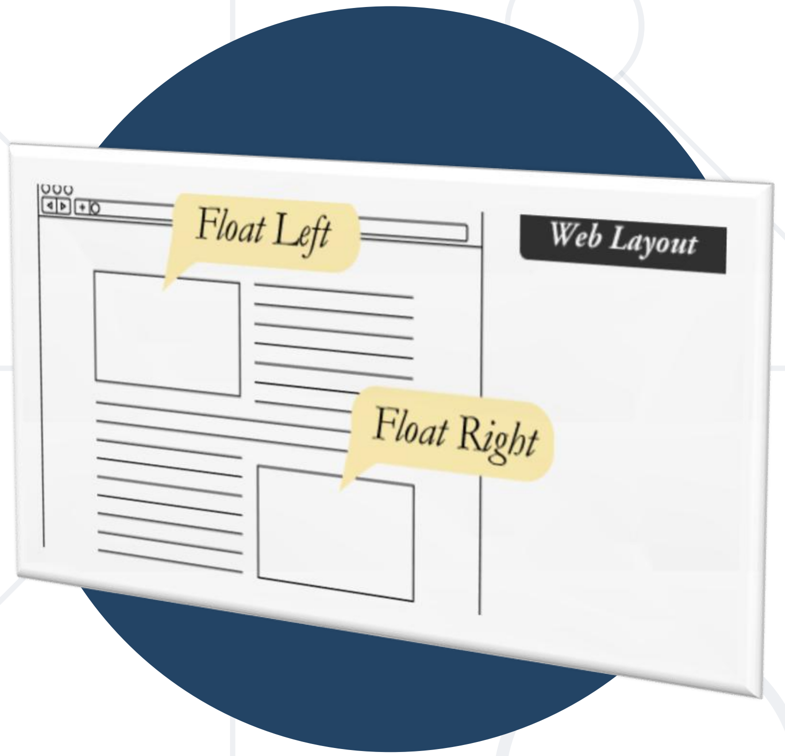
https://softuni.bg

Table of Contents

Have a Question?

sli.do

#HTML-CSS



Float Property

What is Float?

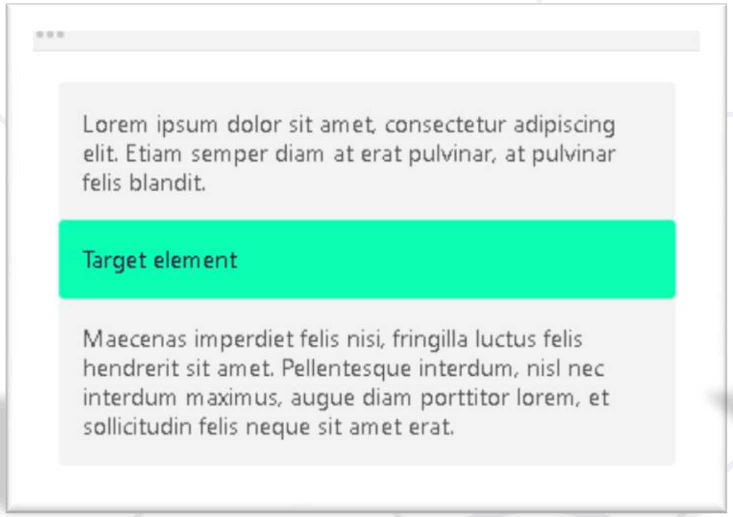
- CSS float is a property that forces any element to float (right, left, none, inherit) inside its parent body with the rest of the element to wrap around it
- The element is removed from the normal flow of the page, though still remaining a part of the flow

Float Property

- Float values:
  - float : **right**; - floats the element to right of it's container
  - float : **left**; - floats the element to left of it's container
  - float : **none**; - it will restrict the element to float
  - float : **initial**; - the element remains to it's default position
  - float : **inherit**; - enables the element to inherit the property from its parent element

Float Property - Example

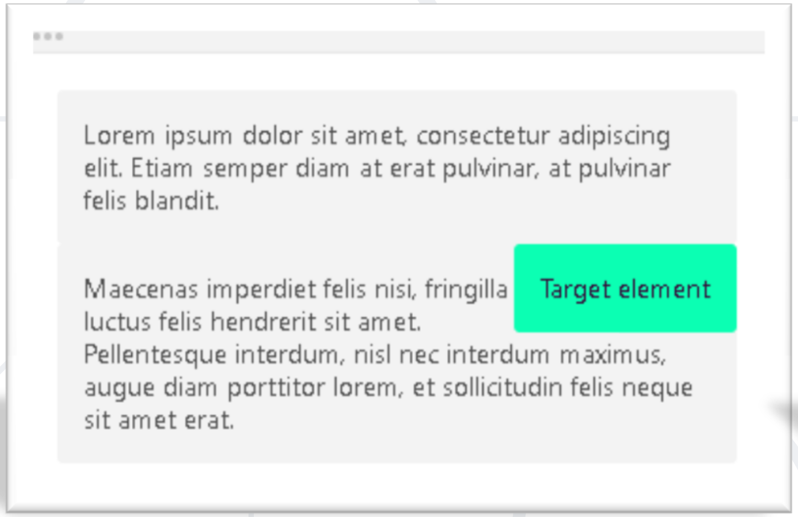
- float: none;


Lorem ipsum dolor sit amet, consectetur adipiscing elit. Etiam semper diam at erat pulvinar, at pulvinar felis blandit.

Target element

Maecenas imperdiet felis nisi, fringilla luctus felis hendrerit sit amet. Pellentesque interdum, nisl nec interdum maximus, augue diam porttitor lorem, et sollicitudin felis neque sit amet erat.

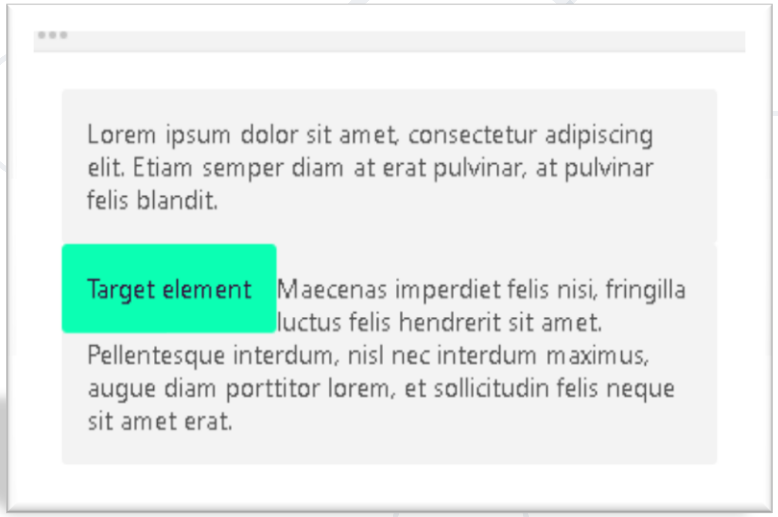
- float: right;


Lorem ipsum dolor sit amet, consectetur adipiscing elit. Etiam semper diam at erat pulvinar, at pulvinar felis blandit.

Target element

Maecenas imperdiet felis nisi, fringilla luctus felis hendrerit sit amet. Pellentesque interdum, nisl nec interdum maximus, augue diam porttitor lorem, et sollicitudin felis neque sit amet erat.

- float: left;


Lorem ipsum dolor sit amet, consectetur adipiscing elit. Etiam semper diam at erat pulvinar, at pulvinar felis blandit.

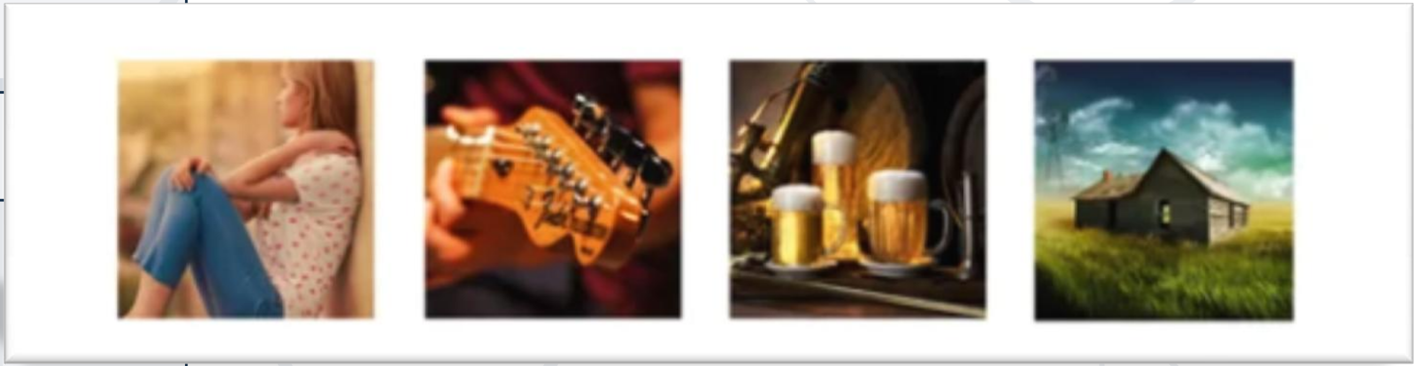
Target element

Maecenas imperdiet felis nisi, fringilla luctus felis hendrerit sit amet. Pellentesque interdum, nisl nec interdum maximus, augue diam porttitor lorem, et sollicitudin felis neque sit amet erat.

Floating multiple Elements

```
<body>
  <img class="image" src="girl.png"/>
  <img class="image" src="music.png"/>
  <img class="image" src="mug.png"/>
  <img class="image" src="nature.png"/>
</body>
```

```
.image {
  float: left;
  width: 150px;
  height: 100px;
  margin: 5px;
}
```



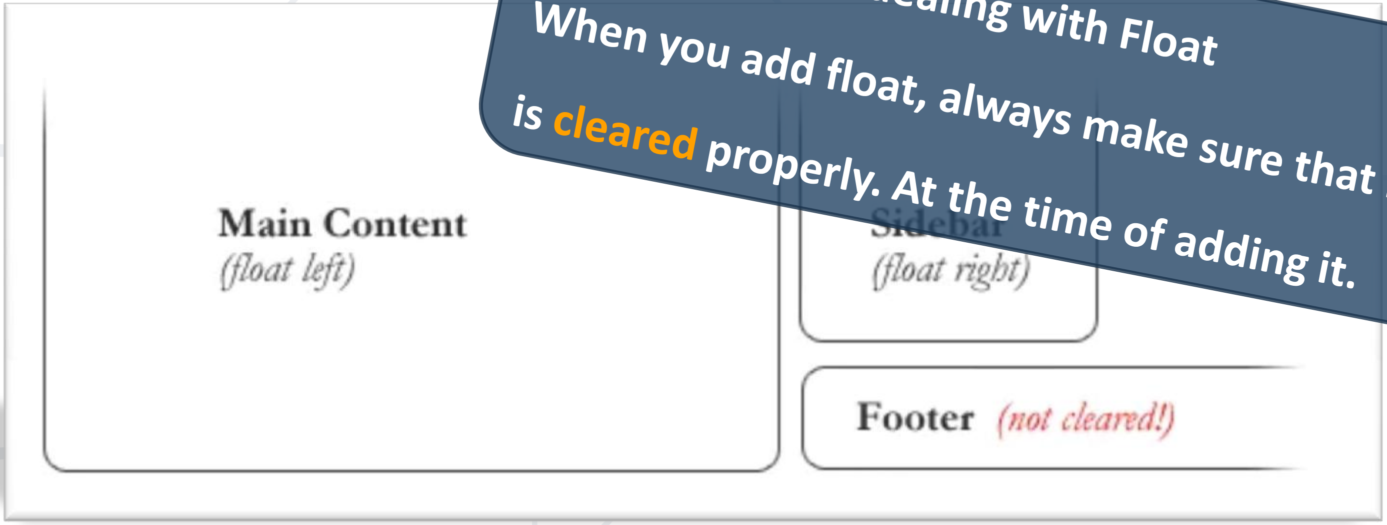
Float Property

- As float implies the use of the block layout, it modifies the computed value of the display values, in some cases:

Specified value		Computed value
inline	⇨	block
inline-block	⇨	block

Clearing the Float

- Float's sister property is **clear**
- An element that has the clear property set on it will not move up adjacent to the float like the float desires but will move itself down past the float



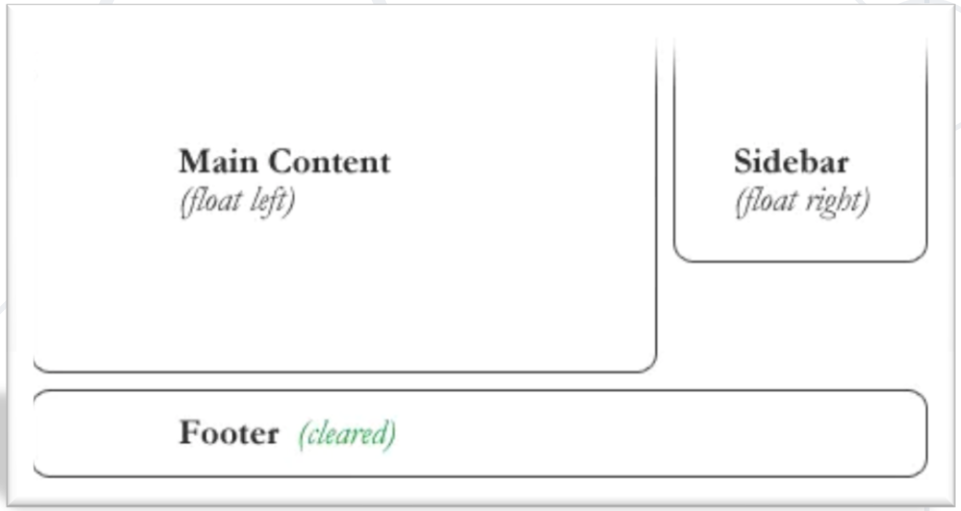
General **rule** for dealing with Float

When you add float, always make sure that it is **cleared** properly. At the time of adding it.

Clearing the Float

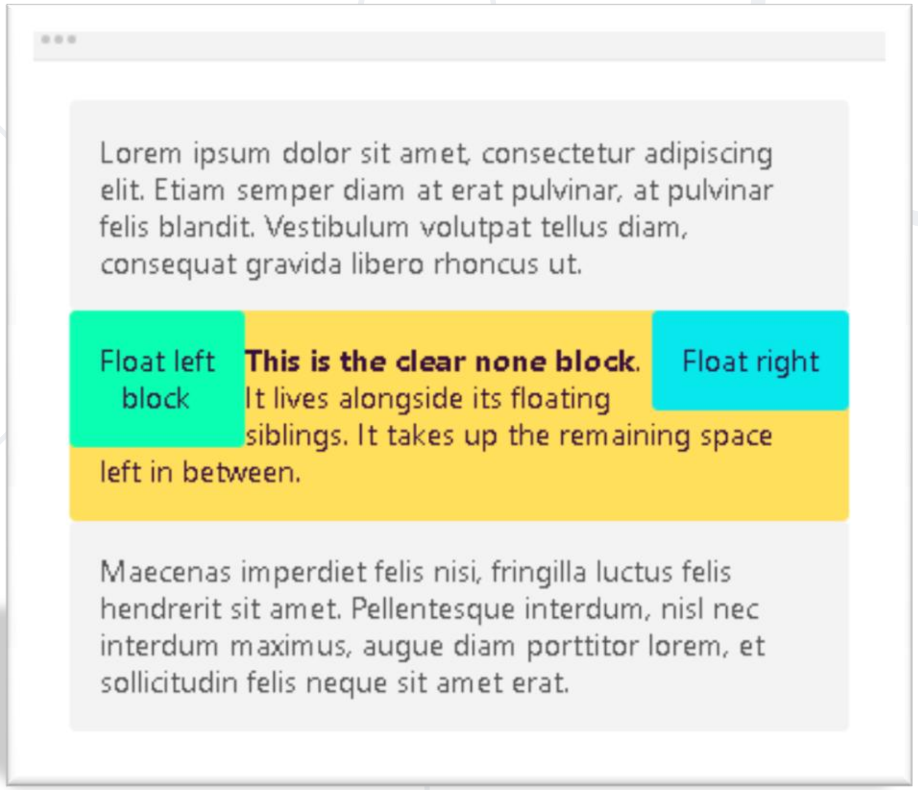
- In the above example, the **sidebar** is **floated** to the **right** and is shorter than the main content area
- The footer then is required to jump up into that available space as is required by the float
- To fix this problem, the **footer** can be **cleared** to ensure it stays beneath both floated columns

```
#footer {
    clear: both;
}
```



Clear

- The clear property has four values:
  - clear: none;
  - clear: left;



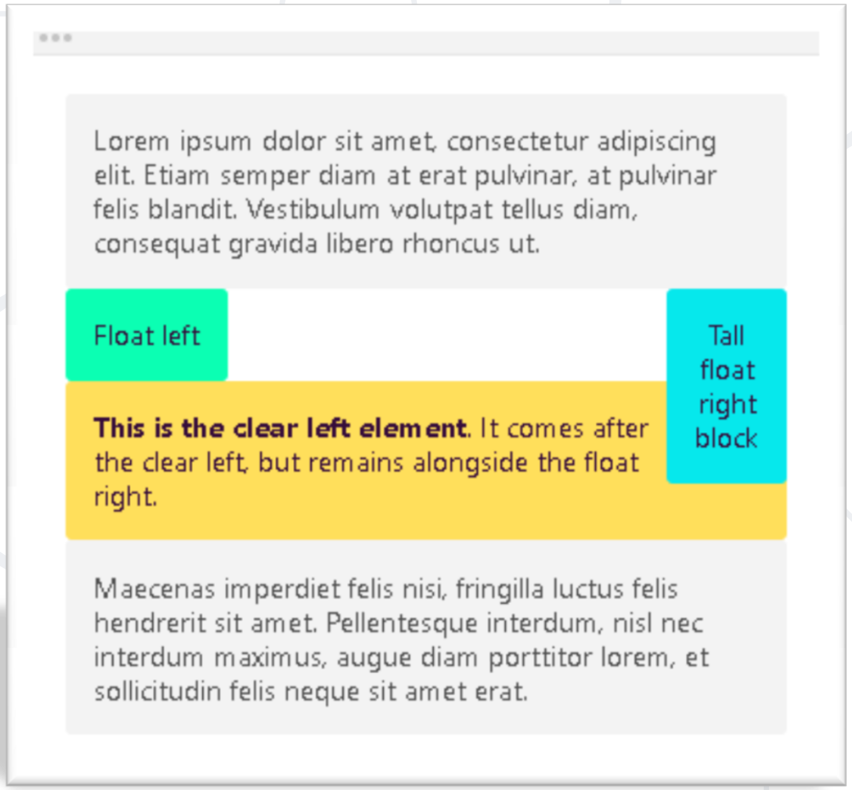
Lorem ipsum dolor sit amet, consectetur adipiscing elit. Etiam semper diam at erat pulvinar, at pulvinar felis blandit. Vestibulum volutpat tellus diam, consequat gravida libero rhoncus ut.

Float left block

This is the clear none block. It lives alongside its floating siblings. It takes up the remaining space left in between.

Float right

Maecenas imperdiet felis nisi, fringilla luctus felis hendrerit sit amet. Pellentesque interdum, nisl nec interdum maximus, augue diam porttitor lorem, et sollicitudin felis neque sit amet erat.



Lorem ipsum dolor sit amet, consectetur adipiscing elit. Etiam semper diam at erat pulvinar, at pulvinar felis blandit. Vestibulum volutpat tellus diam, consequat gravida libero rhoncus ut.

Float left

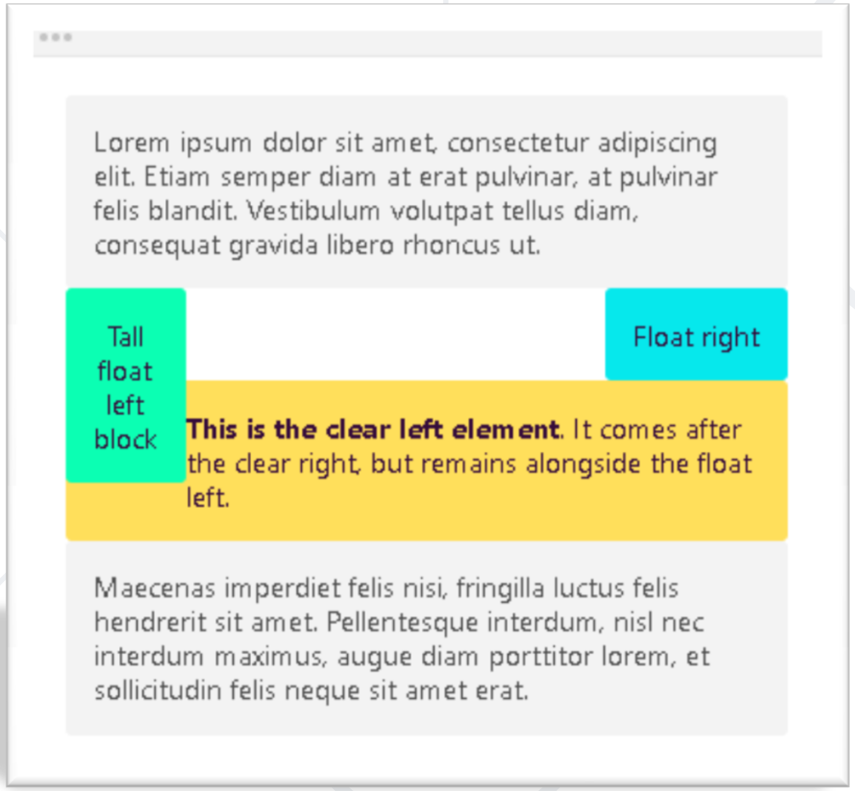
Tall float right block

This is the clear left element. It comes after the clear left, but remains alongside the float right.

Maecenas imperdiet felis nisi, fringilla luctus felis hendrerit sit amet. Pellentesque interdum, nisl nec interdum maximus, augue diam porttitor lorem, et sollicitudin felis neque sit amet erat.

Clear

- clear: right;



Lorem ipsum dolor sit amet, consectetur adipiscing elit. Etiam semper diam at erat pulvinar, at pulvinar felis blandit. Vestibulum volutpat tellus diam, consequat gravida libero rhoncus ut.

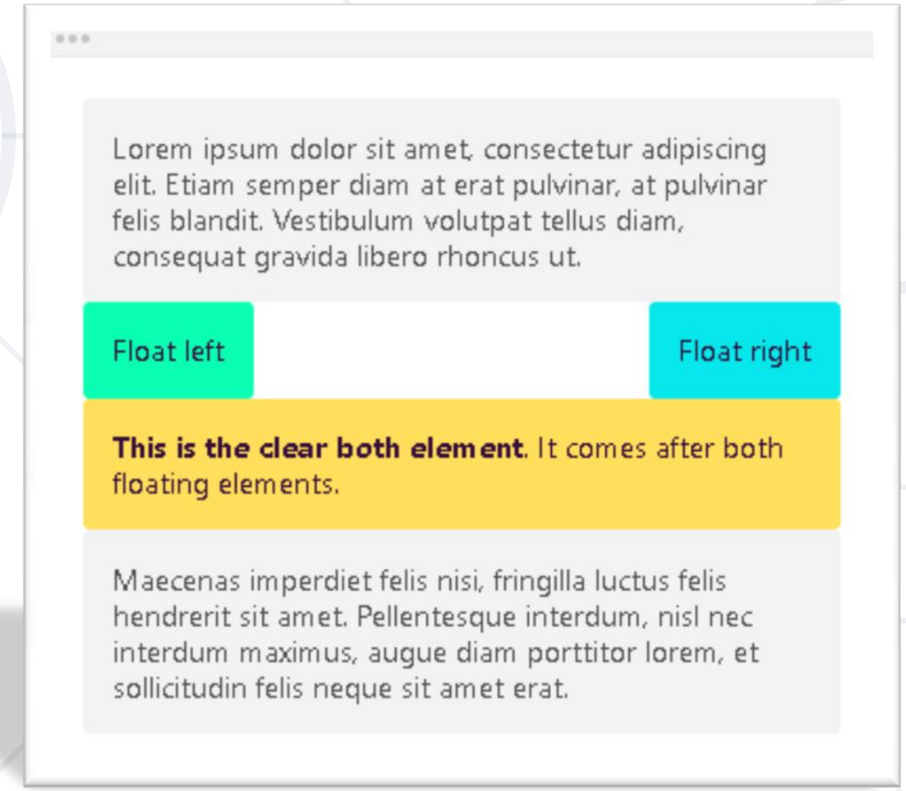
Tall float left block

Float right

This is the clear left element. It comes after the clear right, but remains alongside the float left.

Maecenas imperdiet felis nisi, fringilla luctus felis hendrerit sit amet. Pellentesque interdum, nisl nec interdum maximus, augue diam porttitor lorem, et sollicitudin felis neque sit amet erat.

- clear: both;



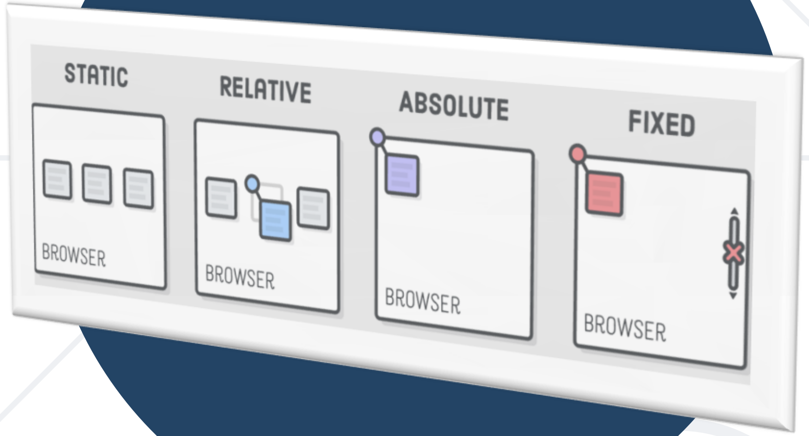
Lorem ipsum dolor sit amet, consectetur adipiscing elit. Etiam semper diam at erat pulvinar, at pulvinar felis blandit. Vestibulum volutpat tellus diam, consequat gravida libero rhoncus ut.

Float left

Float right

This is the clear both element. It comes after both floating elements.

Maecenas imperdiet felis nisi, fringilla luctus felis hendrerit sit amet. Pellentesque interdum, nisl nec interdum maximus, augue diam porttitor lorem, et sollicitudin felis neque sit amet erat.


STATIC
RELATIVE
ABSOLUTE
FIXED
BROWSER
BROWSER
BROWSER
BROWSER

Position

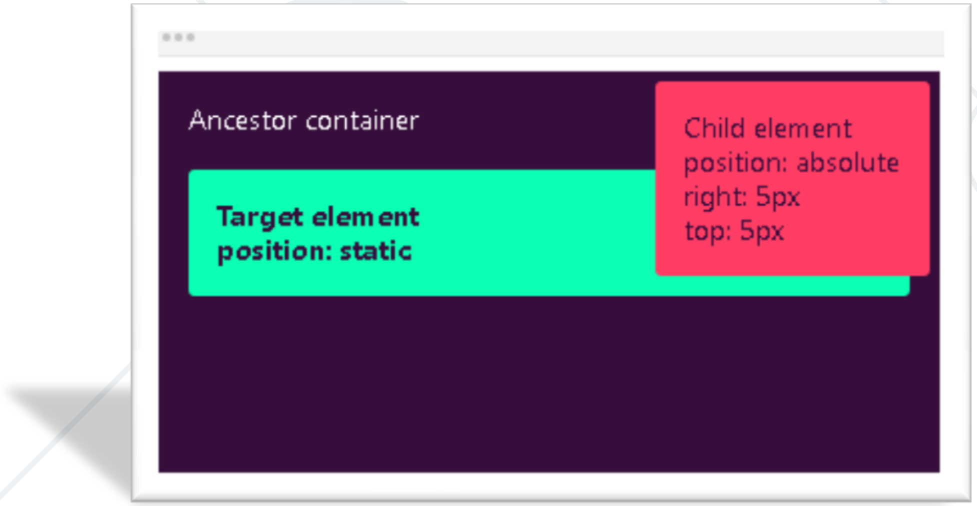
Position

- The **position** property specifies the type of positioning method used for an element
  - Static
  - Relative
  - Absolute
  - Fixed
  - Sticky

Position Static

- Static – the **default** state of every element. It just means "put the element into its **normal position** in the document layout flow"
- Also, it will **not** react to the following properties: top, bottom, left, right, z-index

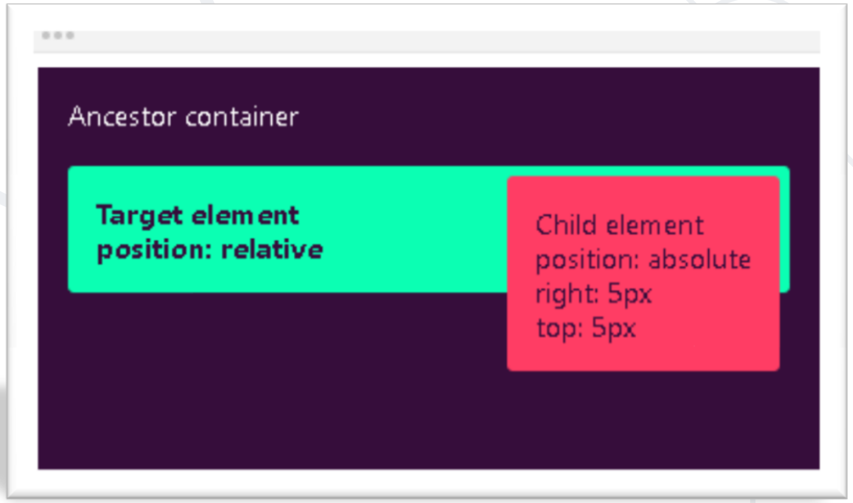
```
[selector] {
    position: static;
}
```



Position Relative

- Relative – very similar to static positioning, **except** that once the positioned element has taken its place, you can then modify its final position with the **positional properties** – left, right, top, bottom, z-index
- It also makes the element **positioned**: it will act as **anchor** point for the absolutely positioned block

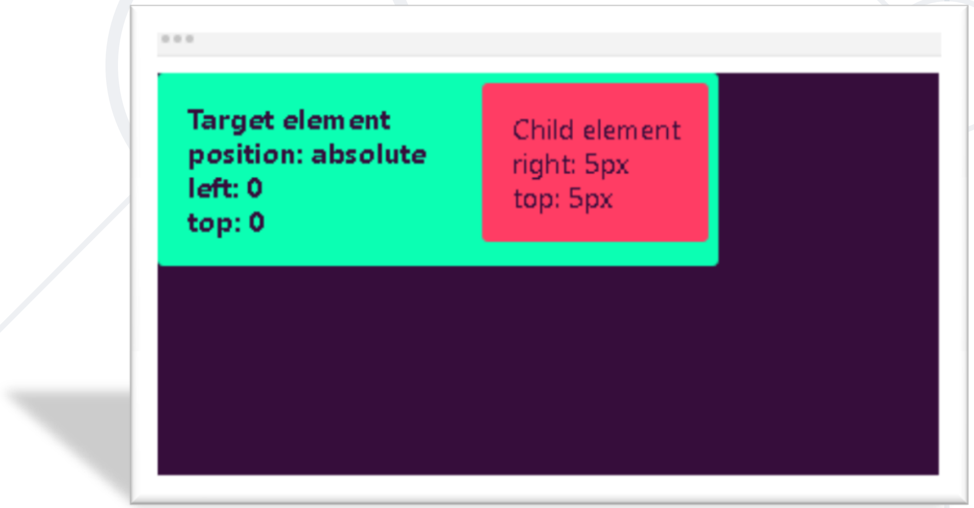
```
[selector] {
    position: relative;
}
```



Position Absolute

- Absolute – the element will **not remain** in the natural flow of the page. It will **react** to the positional properties
- It will positon itself according to the closest positioned ancestor
- Because it’s positioned, it will act as an anchor point for the absolutely positioned block

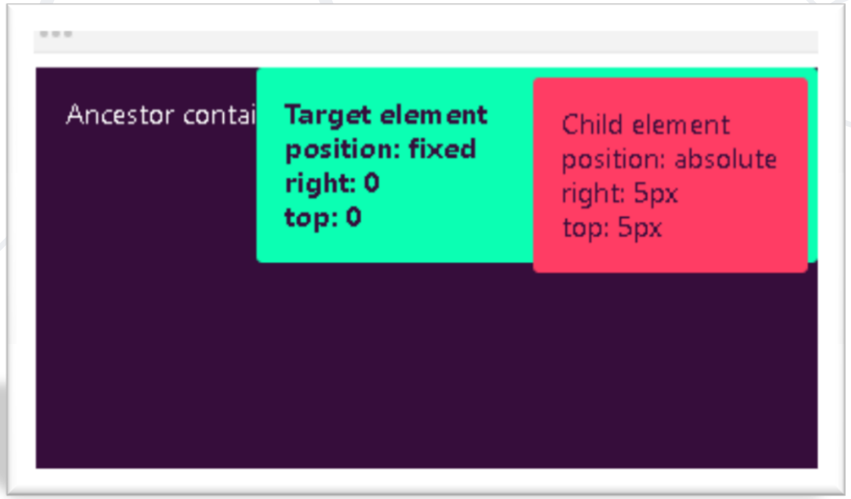
```
[selector] {
    position: absolute;
}
```



Position Fixed

- Fixed – the element will **not remain** in the natural flow of the page. It will **react** to the positional properties
- It will position itself according to the **viewport**
- Because it's positioned, it will act as an **anchor point** for the absolutely positioned block

```
[selector] {
    position: fixed;
}
```



Position Sticky

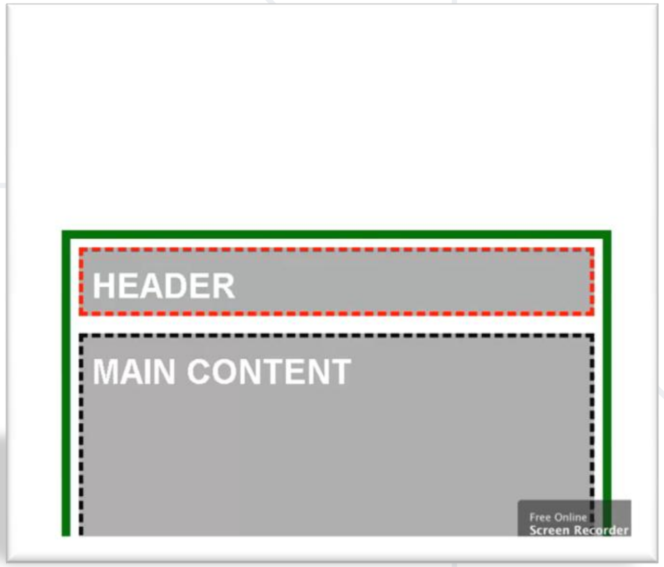
- Sticky – the element is positioned based on the user’s **scroll position**
- A sticky element toggles between **relative** and **fixed**, depending on the scroll position
- It is positioned relative until a given offset position is met in the viewport – then it "sticks" in place (like position: fixed)

Position Sticky - Example

```
.main-container {
   max-width: 600px;
   margin: 0 auto;
   border:10px solid green;
   padding:10px;
   margin-top:40px;
}
.main-header,
.main-content,
.main-footer {
   padding: 10px;
   background:#aaa;
   border: 5px dashed #000;
   margin: 20px 0;
}
```

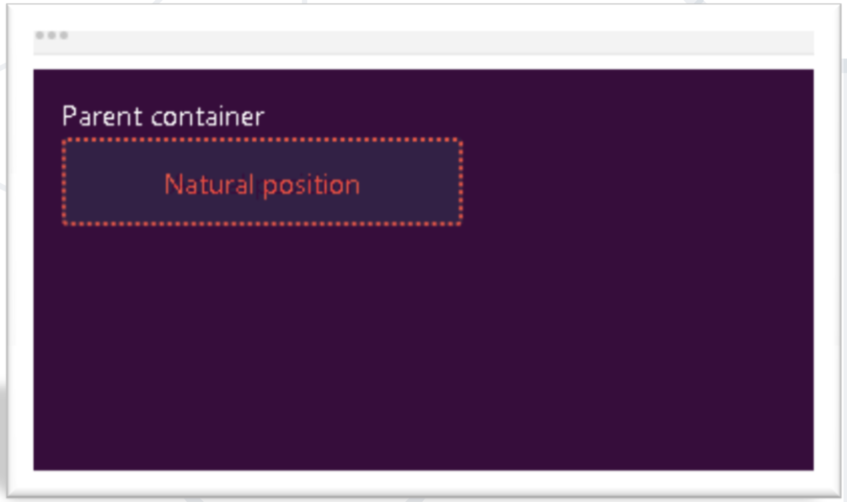
```
<main class="main-container">
        <header class="main-header">HEADER</header>
        <div class="main-content">MAIN CONTENT</div>
        <footer class="main-footer">FOOTER</footer>
</main>
```

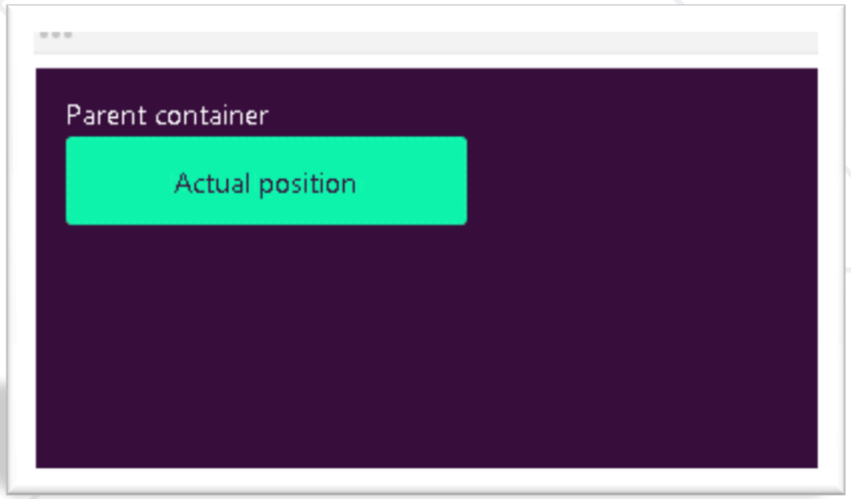
```
.main-content{
   min-height:1000px;
}
.main-header{
   height: 50px;
   border-color: red;
   position: sticky;
   top: 0;
}
```



Bottom

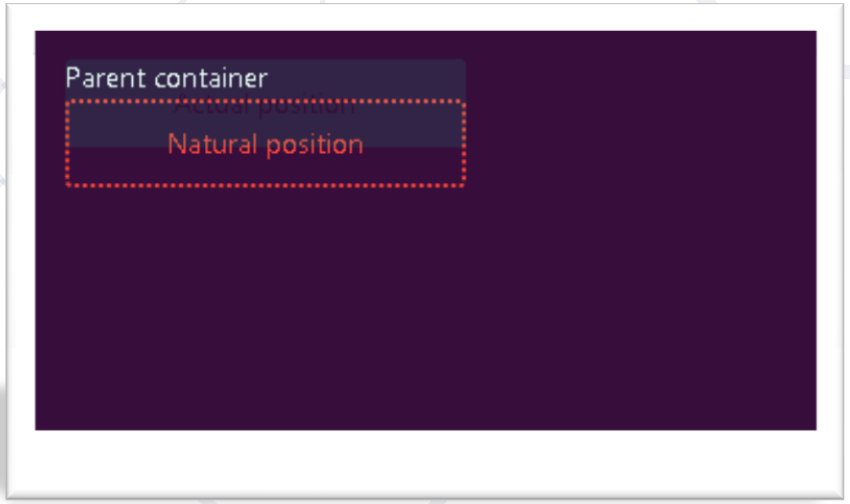
- Bottom – defines the position of the element according to its **bottom** edge
  - bottom: auto; - the element will remain in its **natural** position

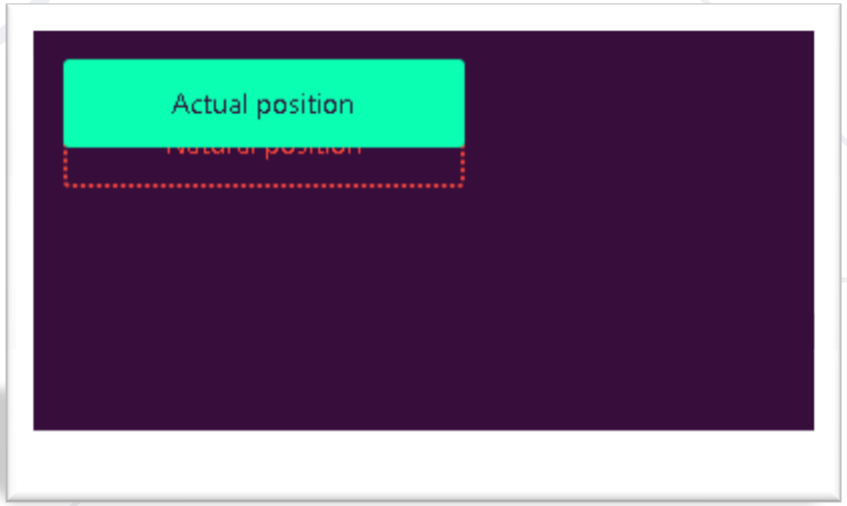




Bottom

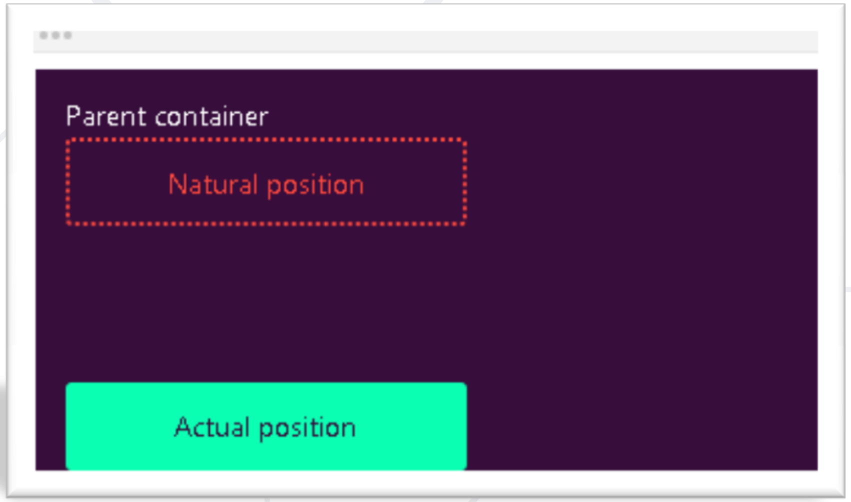
- bottom: 20px;
- If the element is in position **relative**, the element will move *upwards* by the amount defined by the **bottom** value





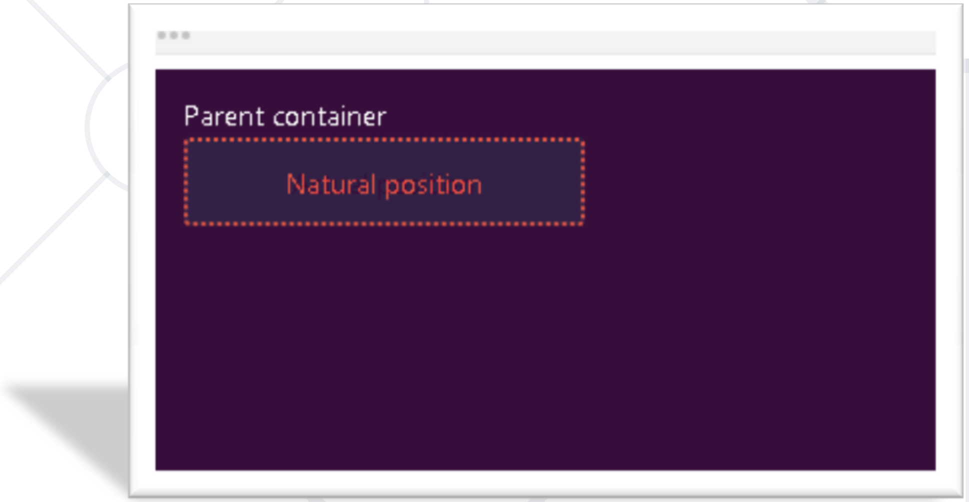
Bottom

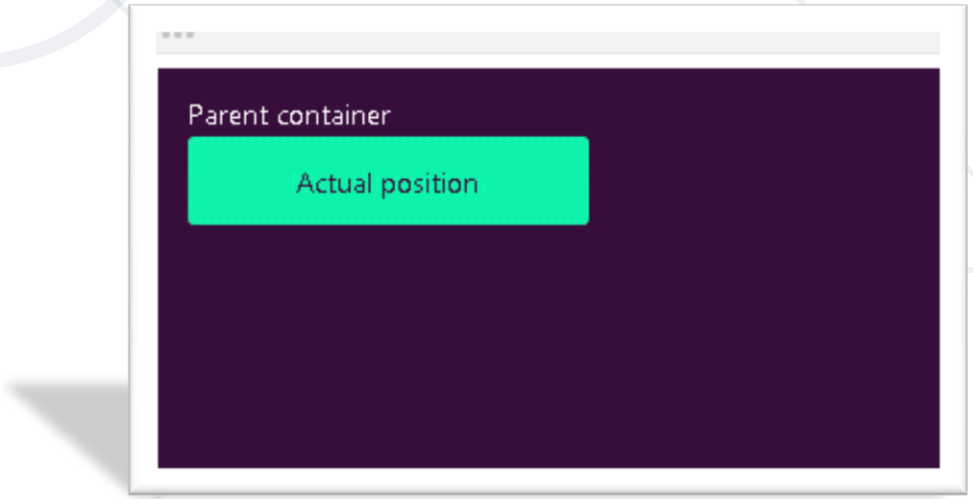
- bottom: 0;
- If the element is in position **absolute**, the element will position itself from the *bottom* of the first positioned **ancestor**



Left

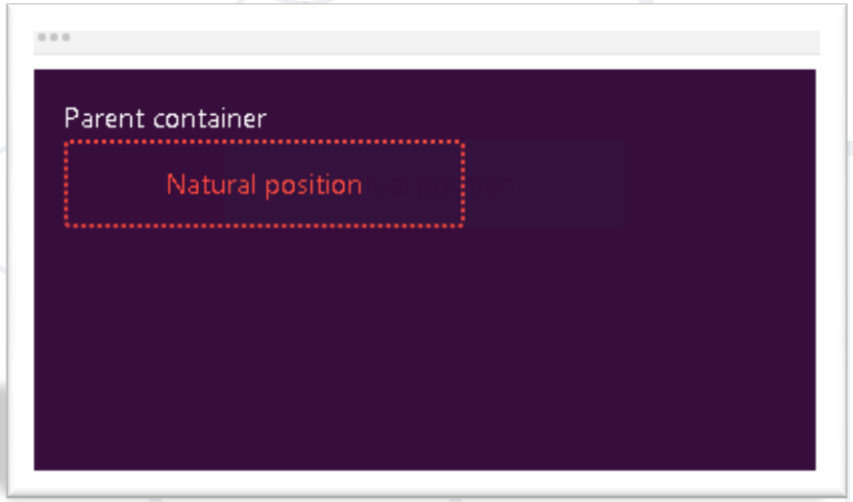
- Left – defines the position of the element according to its **left** edge
  - left: auto; - the element will remain in its **natural** position

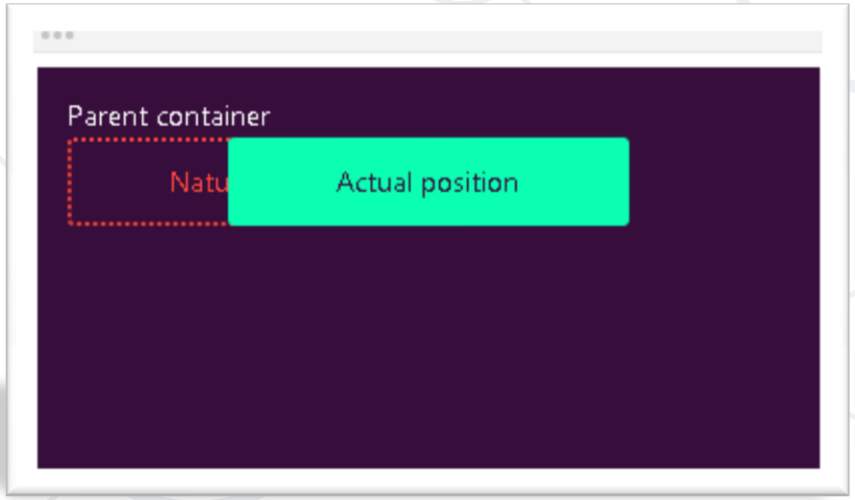




Left

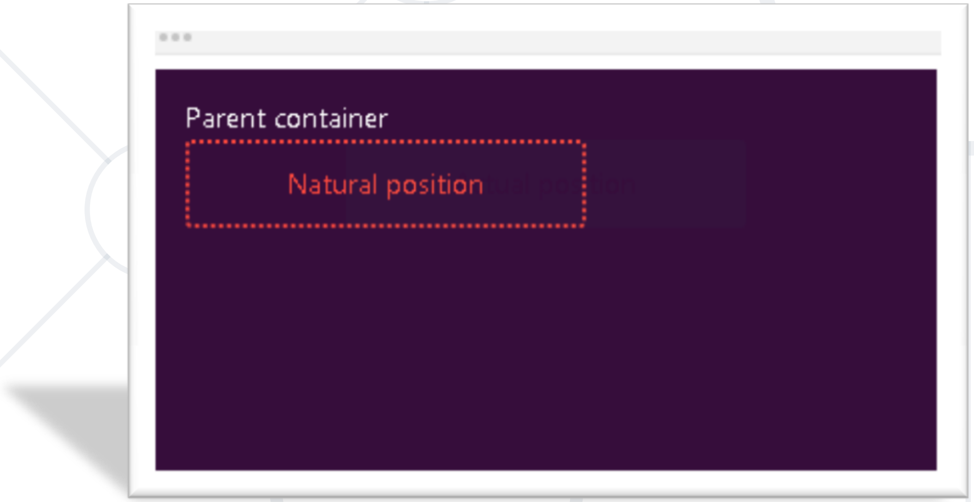
- left: 80px; - if the element is in position **relative**, the element will move *left* by the amount defined by the **left** value

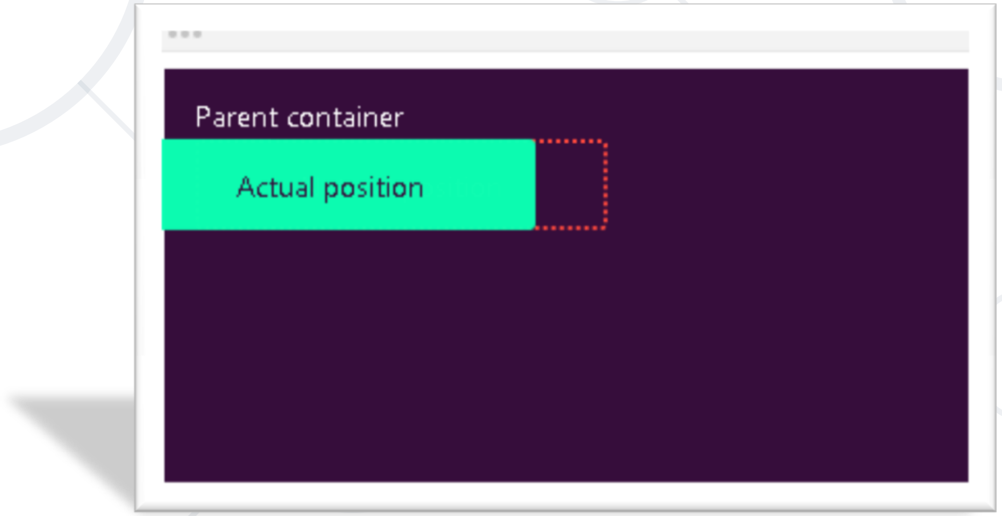




Left

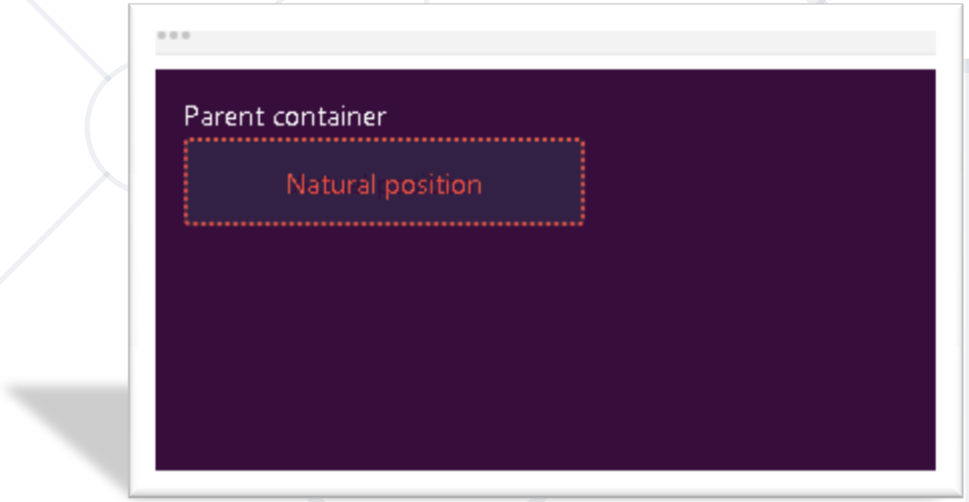
- left: -20px; - if the element is in position **absolute**, the element will position itself from the *left* of the first positioned **ancestor**

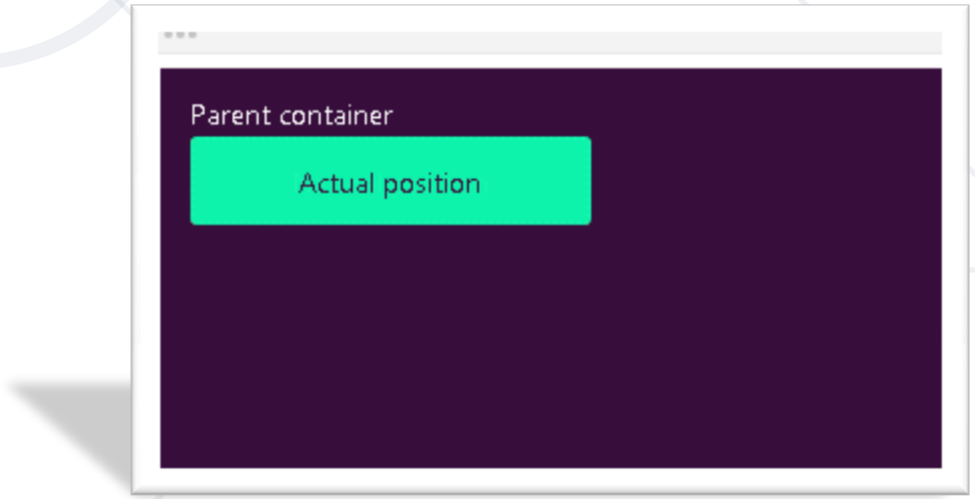




Right

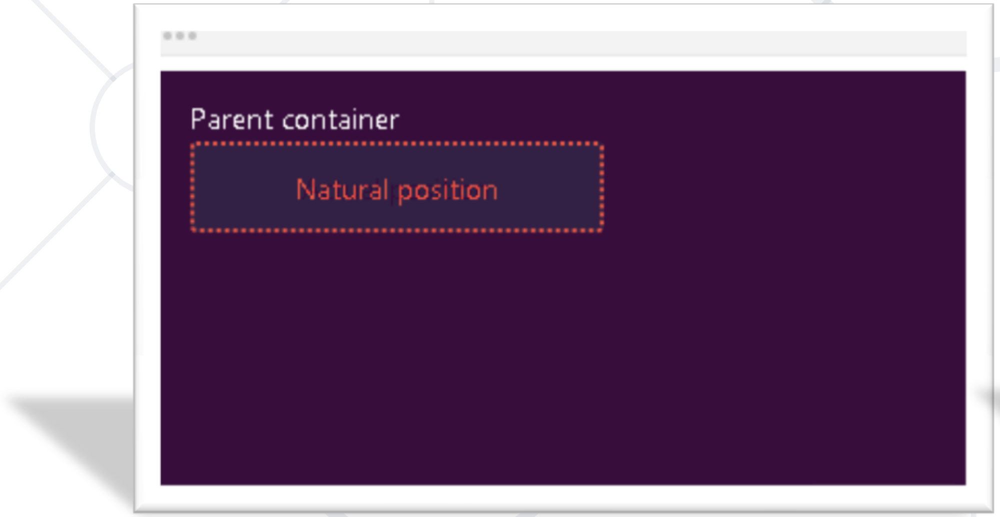
- Right – defines the position of the element according to its **right** edge
  - right: auto; - the element will remain in its **natural** position

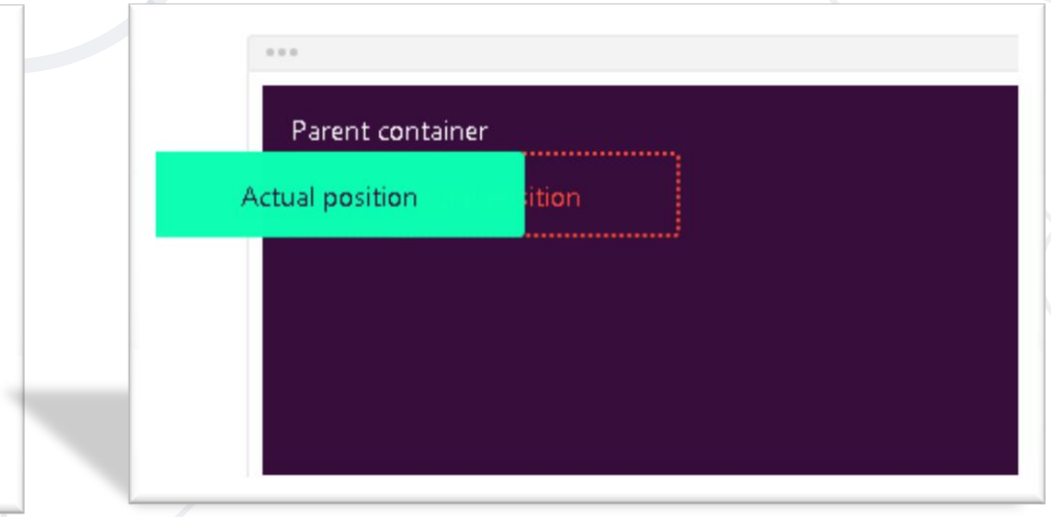




Right

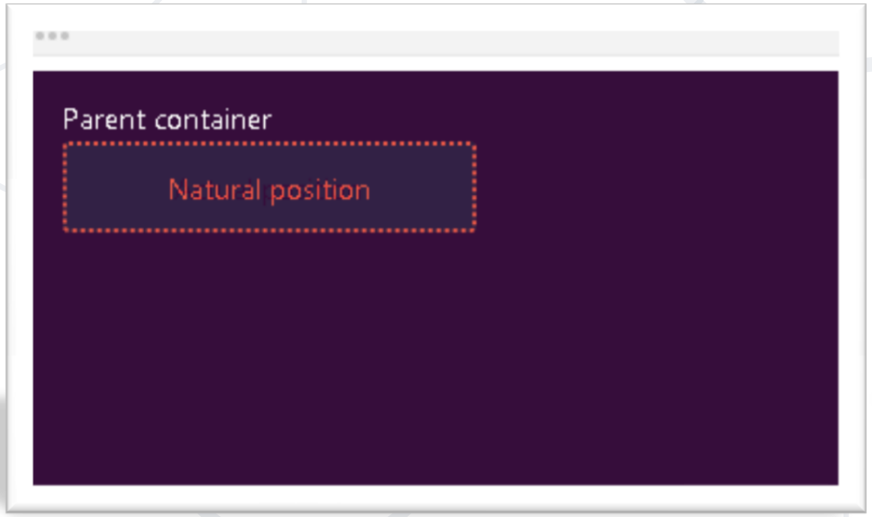
- right: 80px; - if the element is in position **relative**, the element will move *right* by the amount defined by the **right** value

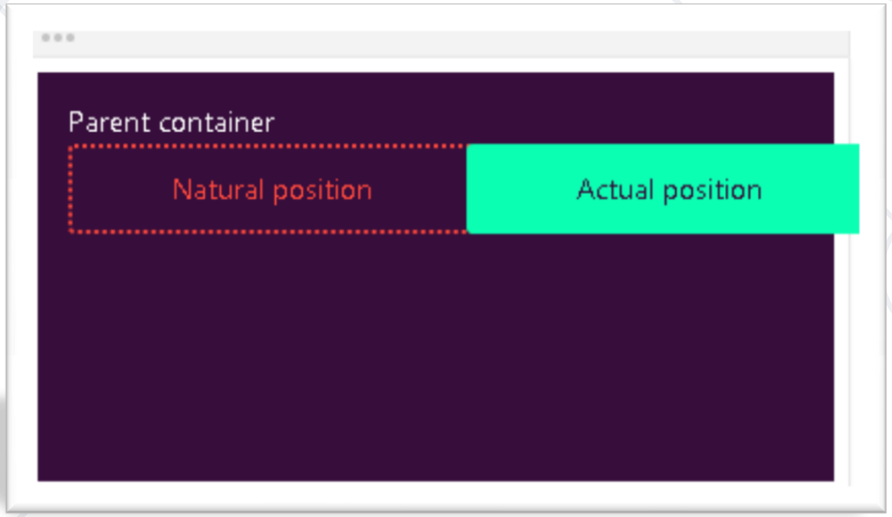




Right

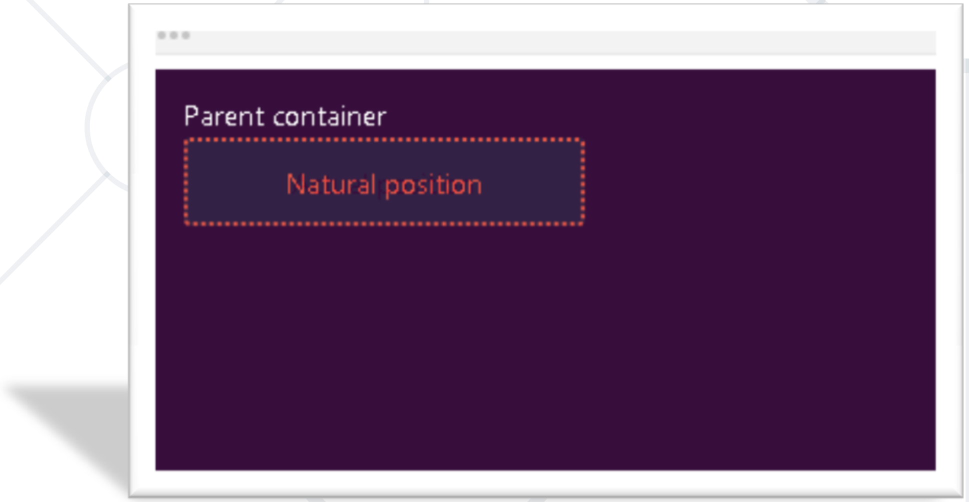
- right: -20px; - If the element is in position **absolute**, the element will position itself from the *right* of the first positioned **ancestor**

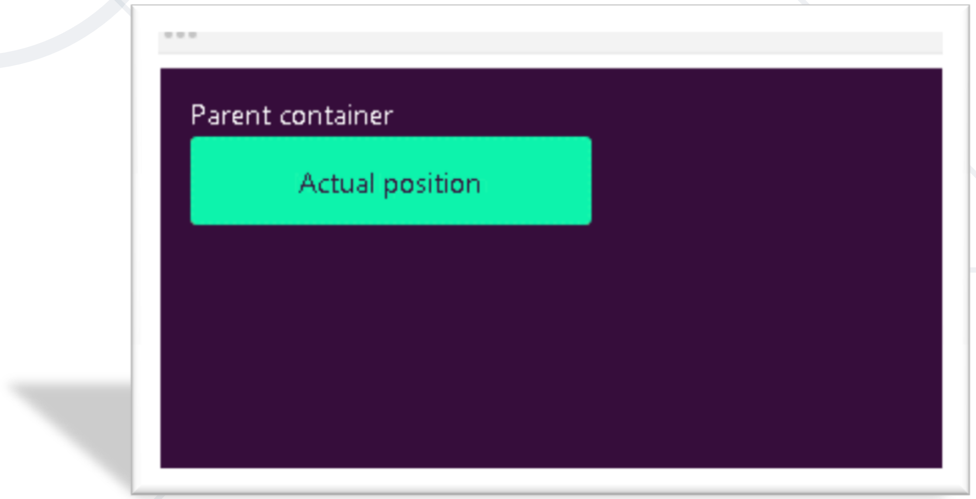




Top

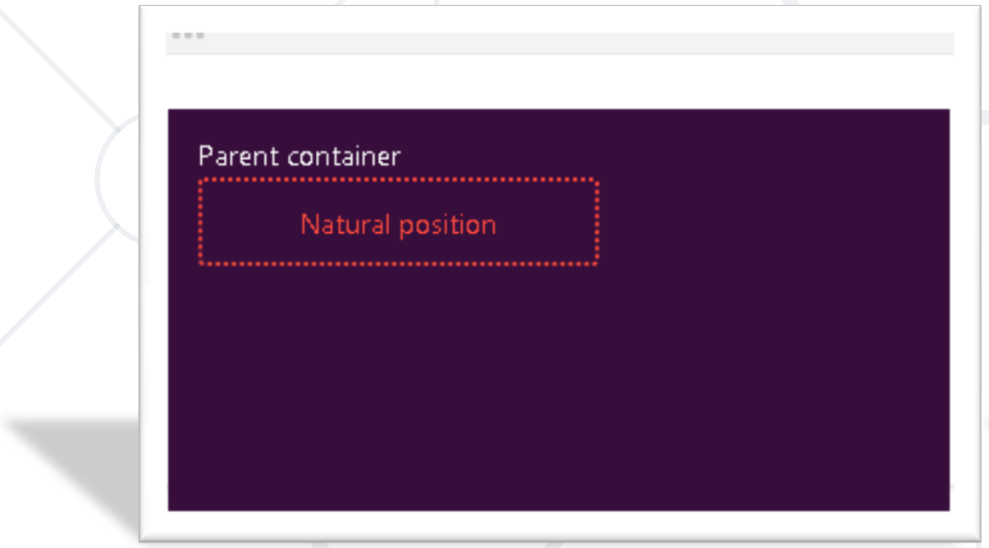
- Top – defines the position of the element according to its **top** edge
  - top: auto; - the element will remain in its **natural** position

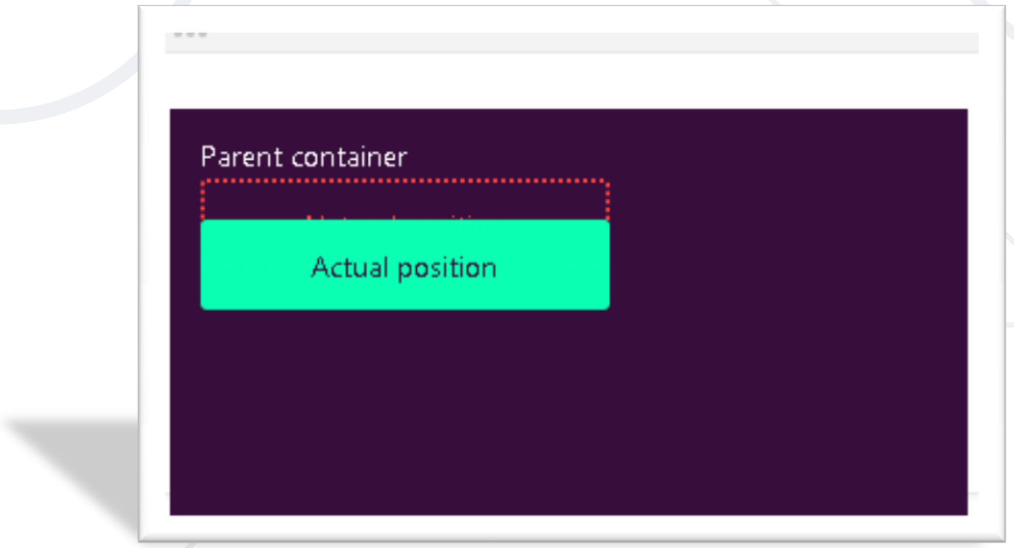




Top

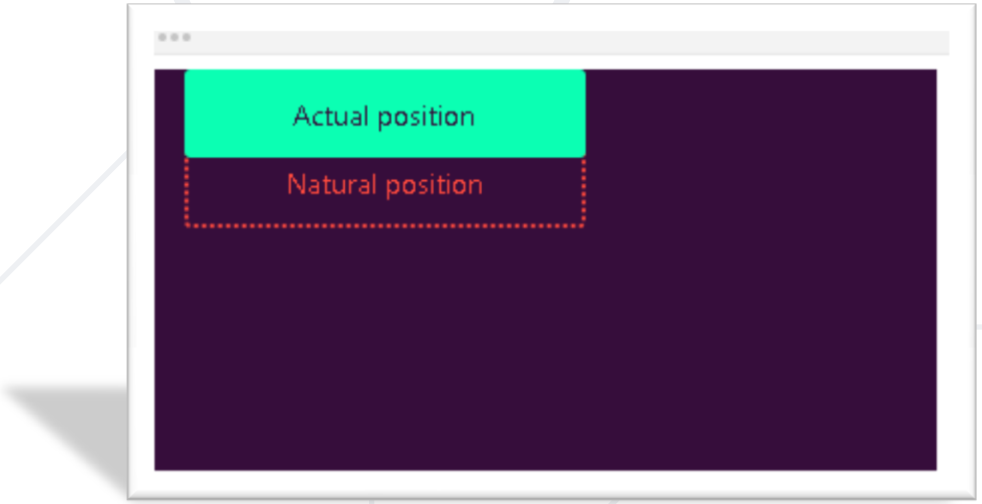
- top: 20px; - If the element is in position **relative**, the element will move *downwards* by the amount defined by the **top** value





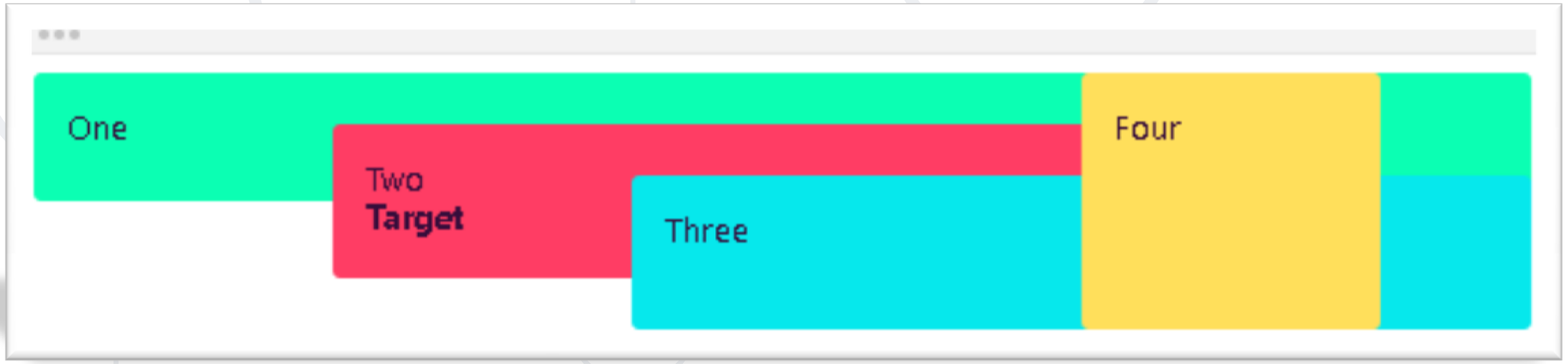
Top

- top: 0; - If the element is in position **absolute**, the element will position itself from the *top* of the first positioned **ancestor**



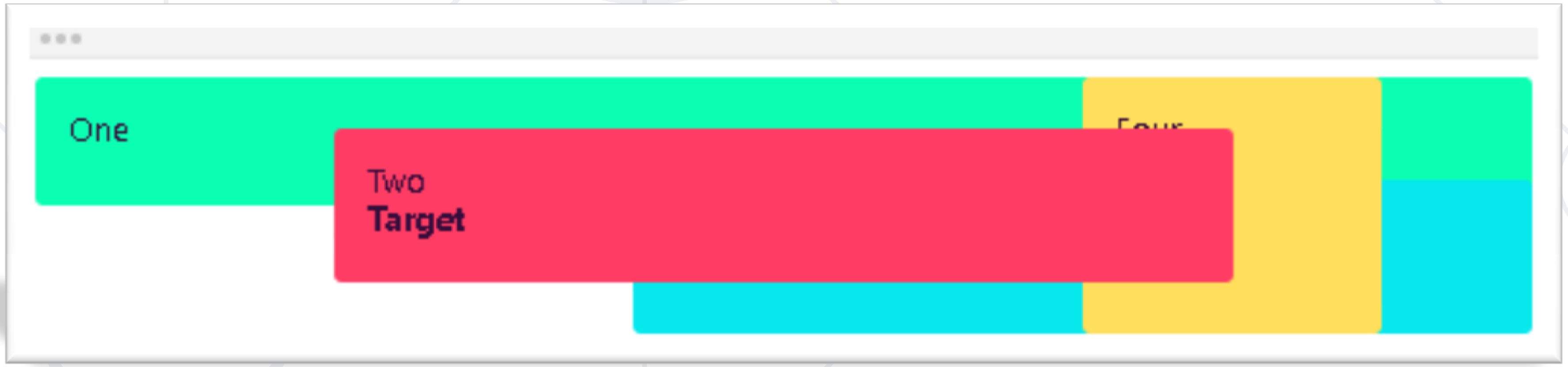
Z-index

- Defines the order of the elements on the **z-axis**. It only works on positioned elements (anything apart from static)
  - **z-index: auto;**
  - The order is defined by the order in the HTML code:
    - First in the code = behind
    - Last in the code = in front



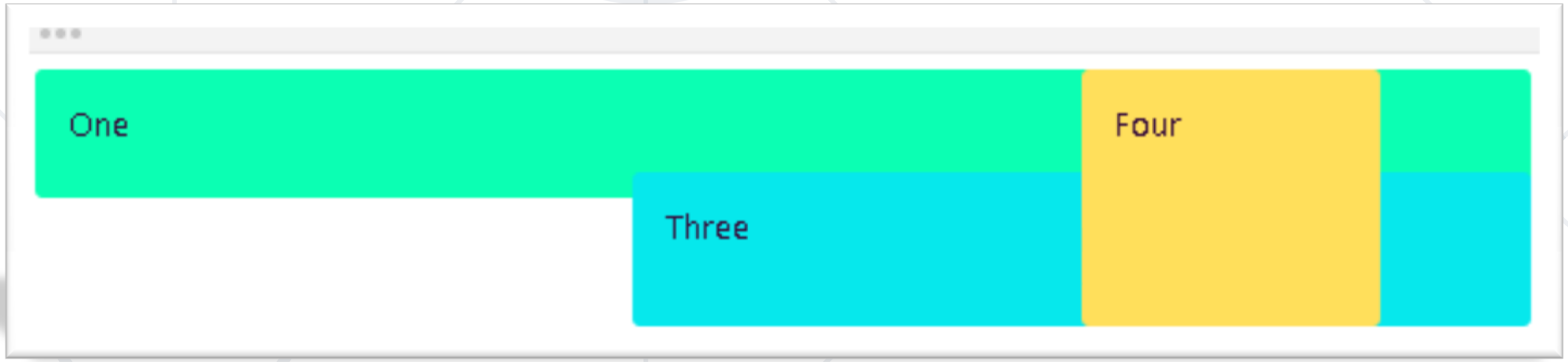
Z-index

- **z-index: 1;**
- The z-index value is relative to the other ones. The target element is move in **front** of its siblings



Z-index

- **z-index: -1;**
- You can use negative values. The target element is move in behind its siblings



Summary

- What is a float?
- How to float elements?
- How to clear float?
- Positioning properties
- How to work with z-index?

SoftUni Diamond Partners

Software University

XSsoftware

SBTech we know sports

telenor

SoftwareGroup *doing it right*

NETPEAK

SmartIT

Postbank

Решения за твоето утре

SUPER HOSTING ®.BG

INDEAVR

Serving the high achievers

INFRAGISTICS®

STEMO®

Computer Systems & Software

SoftUni Organizational Partners

Software University

IS ИНФОРМАЦИОННО ОБСЛУЖВАНЕ

OneBit SOFTWARE

Lukanet.com

WORLD OF MYTHS

Questions?

SoftUni

Software University

SoftUni Svetlina

SoftUni Creative

SoftUni Digital

SoftUni Foundation

SoftUni Kids

License

Trainings @ Software University (SoftUni)

- Software University – High-Quality Education, Profession and Job for Software Developers
  - softuni.bg, softuni.org
- Software University Foundation
  - softuni.foundation
- Software University @ Facebook
  - facebook.com/SoftwareUniversity
- Software University Forums
  - forum.softuni.bg